Attorney's Docket No.: 10559-296001 / P9307

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: RENDERING A PENCIL-SKETCH IMAGE

APPLICANT: ADAM T. LAKE, MARC S. BLACKSTEIN, CARL S. MARSHALL, and DANIEL JOHNSTON

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL227256393US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

11/7/00
Date of Deposit

M.E. Augustine
Signature

Mike Augustine
Typed or Printed Name of Person Signing Certificate

RENDERING A PENCIL-SKETCH IMAGE

TECHNICAL FIELD

This invention relates to rendering a pencil-sketch image from three-dimensional (3D) data.

BACKGROUND

A pencil-sketch image approximates shading and depth by varying the placement and density of discrete line segments. Unlike traditional "smooth", or Gouraud, shading where transitions between light and dark regions of an image are gradual, pencil-sketching uses hard edge boundaries between regions. That is, transitions between regions are created by terminating line segments in the regions, not by blending one neighboring region into another region.

DESCRIPTION OF THE DRAWINGS

Fig. 1 is a view of a Gouraud-shaded 3D model.

Fig. 2 is a wireframe view of polygons in the 3D model.

Fig. 3 is a view of one of the polygons.

Fig. 4 is a view of light hitting the polygon.

Fig. 5 is a flowchart of a process for generating a pencil-sketch image from the polygon.

Figs. 6 to 10 shows textures used for the pencil-sketch image.

Fig. 11 shows a background for the pencil-sketch image.

Fig. 12 is a view showing how a perspective of a 3D model is mapped onto a two-dimensional (2D) surface.

Fig. 13 shows two pencil-sketch images rendered by the process of Fig. 5.

Fig. 14 is a block diagram of a computer system on which the process of Fig. 5 may be implemented.

DESCRIPTION

Fig. 1 shows a 3D model 10. 3D model 10 is a Gouraud-shaded model defined by 3D data. As shown in Fig. 2, the 3D data defines interconnecting polygons 11, which comprise 3D model 10. Polygons 11 are triangles in this embodiment; however, other types of polygons may be used to construct the 3D model. Groups of polygons are organized into meshes, each of which corresponds to an element of the 3D model.

Referring to Fig. 3, the 3D data for a polygon 13 is comprised of coordinates for three vertices 15a, 15b and 15c positioned in Cartesian XYZ (or other) space. These vertices define a face 16 and edges 17a, 17b and 17c for the polygon.

A unit normal vector ("normal") 20a, 20b and 20c at each respective vertex 15a, 15b and 15c affects how the vertex is perceived relative to a predefined reference point (the "eyepoint") 23 (Fig. 4) in the "virtual world" that 3D model 10 inhabits. Taking vertex 15b as an example in Fig. 4, normal 20b determines the amount of light that reaches vertex 15b from a predefined light source 24 in the virtual world. The amount of light is determined using the dot product of unit normal 20b and a unit vector 25 from the light source. The dot product value defines the cosine of angle 18 between the light and the normal. The shading applied to each polygon face is determined based on this angle, as described below. Coordinates for the normals may be stored with the 3D data for each vertex. Alternatively, a normal may be computed "on-the-fly" during pencil-sketch image rendering.

Fig. 5 shows a process 27 for rendering pencil-sketch images from a 3D model. There are two phases to process 27: a pre-processing phase 29 and a run-time phase 30.

In pre-processing phase 29, process 27 obtains (51) a set of "pencil" markings. The pencil markings are bitmap images of line segments that may be scanned-in, read from a disk, retrieved from memory, or generated dynamically. The pencil markings may be straight, curved, or crooked. Also, the

pencil markings may be of varying thickness and length, depending upon the type of textures that they are used to construct.

Process 27 constructs (52) a set of N (N≥1) 2D textures by selecting pencil markings and arranging them uniformly to create various texture maps/tiles. The pencil markings are arranged at different densities and are parallel and/or perpendicular to one another to create different textures. Figs. 6 to 10 shows different types of textures that were constructed by arranging pencil markings.

In Fig. 6, the pencil markings are arranged at a low density and only in the Cartesian X-coordinate direction. Figs. 7 and 8 show higher density versions of the texture shown in Fig. 6. In Fig. 9, the pencil markings are arranged in both the Cartesian X and Y directions (i.e., the pencil markings are cross-hatched) and at a relatively high density. Fig. 10 shows a higher-density version of the texture of Fig. 9. More, less and/or different textures may be used with process 27. For example, a blank texture, which includes no pencil sketch markings, may be used. Since the textures are tiled, the textures may be constructed so that there is continuity between the end point of a line segment on one tile and the start point of a line segment on an adjacent tile.

Thus, when creating the line segments, it is preferable to ensure that the C_0 continuity property holds, where the C_0 continuity property is defined as having the tangent vectors of two curve segments be equal (in both direction and magnitude) at the segments' joint (or intersection) point.

However, this does not always alleviate the appearance of tiling; accordingly, the line segments may be created to ensure that the C_1 continuity property holds, in which the first derivatives (slopes) of the segments at the start and end points of adjacent tiles are roughly equal. This can be difficult to achieve, but can be simulated by randomly selecting the starting point for a line segment and wrapping the line segment around the texture at the end of the tile.

Pre-processing phase 29 set forth above may be performed at any time prior to run-time phase 30. It is noted that a single pre-processing phase may be used to store textures for several different run-time phases.

In run-time phase 30, process 27 selects (53) a background onto which a pencil-sketch image is to be rendered. The background may be selected from a set of backgrounds stored in memory or it may be obtained from another source, such as a disk or a scanned image. The background is an orthographic projection of a relatively large quadrilateral

texture mapped with a paper (or other) texture. Fig. 11 shows an example of a background; however, other backgrounds, or even no background, may be used with process 27.

When rendering a pencil-sketch image from 3D polygon data, process 27 determines (54) which pencil-sketch texture to use for the polygon. Process 27 does this based on the way that the polygon is illuminated, i.e., based on the light that hits the polygon. To determine how light hits a polygon, process 27 obtains (55) a texture value using the vertex normals (see Fig. 3). For polygon 13 (Fig. 4), process 27 calculates the vector dot product of unit normal vector 20b ($\underline{N}$) and unit light vector 25 ($\underline{L}$).

Since $\underline{N}$ and $\underline{L}$ are both unit vectors the product of $\underline{N}\cdot\underline{L}$ is the cosine of the angle 18 formed between the two vectors. If the angle between $\underline{N}$ and $\underline{L}$ is small, then the diffuse component of smooth shading is high and $\underline{N}\cdot\underline{L}$ will have a value close to one. On the other hand, if the angle is large, then the amount of diffuse component in smooth shading is low and $\underline{N}\cdot\underline{L}$ has a value close to zero.

Process 27 takes the maximum of the resulting dot product ($\underline{N}\cdot\underline{L}$) and zero, i.e., $\mathrm{Max}(\underline{N}\cdot\underline{L},0)$ and defines that value as the texture value for the vertex, in this case vertex 20b of polygon 13. The maximum is taken to discount polygons that

are in the back of the 3D model relative to the light source and, thus, produce a negative $\underline{N}\cdot\underline{L}$ value.

For each vertex 20a, 20b and 20c of polygon 13, process 27 obtains (55) a texture value. Process 27 classifies (56) the polygon based on the obtained texture values. Process 27 uses the texture values to associate each vertex of polygon 13 with one of M ($M \geq 1$) bins in memory, each of which corresponds to a predetermined range of values. For example, a system might include three bins having intervals of [0,a], (a,b] and (b,1], where "a" and "b" are adjustable values with $a<b$, $0 \leq a$ and $b \leq 1$, and where square brackets indicate exclusion and parenthetic brackets indicate exclusion, e.g., "a" is included in the range [0,a] but excluded from the range (a,b]. So, in this example, if a texture value of vertex 20b is "a", vertex 20b will be associated with bin [0,a]. Different numbers and/or ranges of bins may be used in process 27.

Process 27 associates (57) one of the N pencil sketch textures from Figs. 6 to 10 with polygon 13 based on the classifications of the polygon's vertices. Process 27 builds n ($n \geq 1$) face lists in memory, each of which corresponds to one of the N textures ("N" here is not necessarily equal to "n"), and assigns polygon 13 to one of those face lists based on the bins into which the polygon's vertices fall. For polygon 13,

if each vertex 20a, 20b and 20c falls in the same bin, the polygon is appended to a face list that correlates to the bin. If different vertices of polygon 13 fall into different bins, then the polygon is appended to the most appropriate face list. For example, if two vertices belong to the same bin, but one other vertex does not, the polygon may be appended to the face list for that bin despite the other vertex.

Once process 27 determines (54) the texture for polygon 13, process 27 projects (58) polygon 13 onto a 2D surface. Referring to the example shown in Fig. 12, this is done by determining the XY coordinates on 2D surface 30 (e.g., a computer monitor) of a polygon 31 on 3D model 32. Process 27 projects the coordinates of the polygon onto 2D surface 30, resulting in a 2D representation of the polygon.

Referring back to Fig. 5, process 27 maps (59) the appropriate texture onto the 2D representation of polygon 13. As noted, the texture of polygon 13 is determined based on the face list to which polygon 13 is appended. Process 27 is repeated for each polygon in a 3D model, resulting in a pencil-sketch image of the 3D model. Examples of pencil-sketch images generated by process 27 are shown in Fig. 13.

Process 27 may be used to create animation cels for cartooning. For example, a 3D model, such as model 10, may be

generated, and then positioned in a desired manner. Process 27 may be executed on the model to produce a pencil-sketch 2D image for that position. Then, the 3D model 10 can be re-positioned (e.g., rotated), and process 27 executed on the re-positioned model to produce a pencil-sketch 2D image for a different perspective of the model. This process may be repeated to produce pencil-sketch 2D images for any number of model positions. Thus, process can generate animation cels automatically, meaning without the use of hand-drawn sketches.

Process 27 runs in real-time, which facilitates the animation process. That is, in conventional hand-drawn animation, artists cannot interactively change the appearance/view of a character without re-drawing the character manually. Process 27 permits this because it renders frames of animation (i.e., 2D images) dynamically and automatically for a given viewpoint in real-time. In this regard, the viewpoint is not the only aspect of a frame that can be dynamically manipulated using process 27. Light moving relative to a character and model changes the locations of shadows on those objects, just as in a conventional 3D Gouraud-shaded scene.

Process 27 can be used for interactive technical illustrations and real-time video game play. For example, a

pencil-sketch game may be constructed in which a user navigates throughout a virtual world that appears in 2D, e.g., a world that simulates a newspaper comic. So-called "How-To" manuals, particularly the online variety, often make use of pencil-sketch drawings to illustrate aspects of a model. Process 27 may be used to allow a reader to examine the model from different angles/perspectives.

Fig. 14 shows a computer 35 for rendering pencil-sketch images using process 27. Computer 35 includes a processor 36, a memory 37, a storage medium 39 (e.g., a hard disk), and a 3D graphics accelerator card 40 for repositioning a 3D model and manipulating 3D data (see view 41). Storage medium 39 stores 3D data 42 which defines a 3D model, and computer instructions 44 which are executed by processor 36 out of memory 37 to render pencil-sketch images using process 27 and 3D data 42. Memory 37 also stores the face lists and bins noted above.

Process 27 is not limited to use with the hardware and software of Fig. 14; it may find applicability in any computing or processing environment and with any type of machine that is capable of running a computer program. Process 27 may be implemented in hardware, software, or a combination of the two. Process 27 may be implemented in computer programs executing on programmable computers that

each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices. Program code may be applied to data entered using an input device to perform process 27 and to generate output information.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on a storage medium or device (e.g., CD-ROM, hard disk, or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform process 27. Process 27 may also be implemented as a computer-readable storage medium, configured with a computer program, where, upon execution, instructions in the computer program cause the computer to operate in accordance with process 27.

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is: